AIAA Aviation
16-20 June 2014, Atlanta, GA
7th AIAA Flow Control Conference

Optimization of Perturbation Parameters for Simulated Free Shear Layer Flow

Rodney A. Martin
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035
rodney.martin@nasa.gov

Upender K. Kaul
NASA Advanced Supercomputing (NAS) Division
NASA Ames Research Center
Moffett Field, CA 94035
upender.kaul@nasa.gov

Abstract

This paper provides details on the optimization of phase and amplitude of perturbations for simulated free shear layer flows. The goal of the optimization is to maximize or minimize the rate of growth of the shear layer, based upon first-principles physics-based simulations that represent solutions to the fully nonlinear Navier-Stokes equations. These simulations have been obtained using a unique method [1], [2] that considerably reduces the computational burden normally associated with obtaining such solutions. In fact, the development of active flow control methodologies is often based upon reduced order models of the Navier-Stokes equations to avoid this computational overhead. Various regression methods were used to approximate the shear layer thickness as a function of the phase and amplitude of perturbations used to excite the flow dynamics as a proxy for using a simulation based upon first principles, in order to reduce computational burden even further. It was found that nonlinear regression methods overall outperformed linear regression methods, owing to the fundamentally nonlinear nature of the data.

1 Introduction

Free shear layers or mixing layers are ubiquitous in real world applications. For example, they are found in flow reactors, cavity flows, flows over aircraft wings at angle of attack, bluff body wakes, jets, etc. Often, there is a need to optimize the dynamics of such flows. The objective may be to minimize or maximize the shear layer growth rate. In flow reactors, we seek to maximize it and in other applications to minimize it. Maximization of growth rate produces efficient mixing of fluids in flow reactors and thus an efficient reaction mechanism. Minimization of the growth rate suppresses the cavity tones or jet exhaust noise, for example.

Many reduced order models (ROM) [3],[4],[5] have been proposed for mixing layers that can be used to optimize such flows. These ROM are an approximation to the physics of such flows, but they solve the control or optimization problem quickly. On the other hand, first-principles modeling of mixing layers is an exact representation of the physics of such flows, but they take too long if we seek to optimize these flows. First principles modeling requires the solution of Navier-Stokes equations of fluid dynamics that are highly nonlinear. These solutions on present day supercomputers take on the order of hours to days to compute. Therefore they are not a viable route for optimization of these flows in real world applications. But, a unique method [1],[2] has been devised to optimize such flows that is based on first principles. This method uses transformations between spatially growing mixing layers and time-evolving mixing layers that eliminates the need to compute the spatially growing layer directly. Instead, a time-evolving layer is computed that takes orders of magnitude less time. The execution speed of this approach is on the same order as the ROM mentioned earlier. Hence, we focus on this approach in this study to generate some canonical solutions

(data) rapidly and we explore various methods of developing a proxy or a surrogate to model the data thus generated , based on the time domain results.

2 Methodology

A variety of regression methods can be used to obtain the best fit to data generated from the simulations discussed in Sec. 1. The motivation is to use the resulting model as a data-driven proxy; i.e., an equivalent simulation to the physics-based Navier-Stokes equations at a greatly reduced computational burden, similar to what was performed by Pressburger *et al.* [6]. To facilitate selection of the best performing regression method, the NMSE (Normalized Mean Square Error) metric was used. This metric aided hyperparameter selection for the applicable regression methods by offering the ability to optimize the NMSE as an objective function of the hyperparameters. The normalized mean-square error (NMSE) [7] is the mean-square error divided by the variance of data; this allows for comparing regression techniques because it compensates for the difficulty of fitting the data. Low values of NMSE indicate a better fit. It is a traditional metric for assessing the goodness-of-fit of a regression algorithm which governs model fidelity, and is often a better choice than using the RMS (root-mean square) due to the ability to compare regression algorithms more equitably.

Formally, the optimization used for regression can be posed as shown in Eqn. 1, with the NMSE metric as the objective, where the index k represents the time index, y_k represents the actual target parameter value, and $\hat{y}_k$ represents the estimated target parameter value. The target parameter value of interest for our application is the free shear layer thickness at a simulation time corresponding to the first pairing. Any number of optimization techniques can be used to solve this problem, however here a simple grid search was sufficient based upon the limited number of tuples ($P = 102$) used to fit a model.

$$\begin{aligned} &\text{minimize} && J = \frac{\sum_{k=1}^{P}(y_k - \hat{y}_k(\lambda))^2}{\sum_{k=1}^{P}(y_k - \bar{y})^2} \\ &\text{subject to} \\ &\lambda \in \mathcal{S} \end{aligned} \tag{1}$$

where

$$\begin{aligned} P &= \text{number of } (phase, amplitude) \text{ tuples} \\ \bar{y} &= \frac{1}{P}\sum_{k=1}^{P} y_k \\ \hat{y}_k(\lambda) &= f(\mathbf{u}_k, \lambda) \\ \mathbf{u}_k &= \text{Vector of regressors} \\ \lambda &= \text{Regression-specific hyperparameter} \\ \mathcal{S} &= \text{Regression-specific hyperparameter tuning domain} \end{aligned}$$

2.1 ℓ_2 regularized linear regression (Ridge Regression)

One of the regression methods tested was linear ridge regression, in which the associated regularization coefficient was used as a hyperparameter to optimize the MSE as a function of how well conditioned the solution should be. A well-posed or well-conditioned problem is one that yields a solution that meets the criteria of existence, uniqueness, and robustness (e.g. low sensitivity to natural variation in the data). The linear regression techniques to be evaluated are linear in the parameters to be estimated only, however basis functions for the regressors themselves to be studied will involve both affine and quadratic regressors in the same vein as was presented in both [6] and [8]. As such, both linear and quadratic regressors were used, where the use of quadratic regressors include the x_i^2 regressors as well as all $\binom{N}{2}$ quadratic pairs of parameters, x_i and x_j.

2.2 Support Vector Regression (SVR)

In this subsection, a brief description of the $\eta-$support vector regression algorithm used for target prediction is provided, as documented in [9]. Given a finite set of multivariate observations, it is possible to reconstruct an input and target set that takes the form shown in Eqn. 2, where $\mathbf{U} \overset{\triangle}{=} [\mathbf{u}_0 \dots \mathbf{u}_P]^\top$ is an input data matrix of size $(P \times 2)$ and the corresponding output is denoted by $\mathbf{y} \overset{\triangle}{=} [y_0 \dots y_P]^\top$, called the target vector. Thus, there are 2 parameters and P observations. Once the $\eta-$support vector regression algorithm is appropriately trained, it is possible to estimate a target function $f(\mathbf{u}_k)$ that imposes an upper bound of η on the actual number of observed targets $\{y_k\}_{k=0}^{P}$ for all the input data $\{\mathbf{u}_k \in \mathbb{R}^2\}_{k=0}^{P}$.

$$\mathbf{y} = f(\mathbf{U}) \tag{2}$$

The target function $f(\cdot)$ is a linear combination of specific weighted training and test points with an additional offset which is often known as bias, ρ. The chosen training instances with m non-zero weights are called *support vectors* (SVs, $\mathbf{u}_i$) and they are the statistically sufficient representatives of the model. This implies that given the model, any training points not associated with SVs can be discounted without changing the performance of the algorithm. The target function is shown in Eqn. 3.

$$f(\mathbf{u}_k) = \sum_{i=1}^{m} (\alpha_i - \hat{\alpha}_i) \langle \mathbf{u}_i, \mathbf{u}_k \rangle + \rho \tag{3}$$

The support vectors and their corresponding weights, α_i and $\hat{\alpha}_i$ result from the solution of a quadratic programming optimization problem in dual form. The expression of the primal problem is shown in Eqn. 4. Further details on the cost function and optimization problem can be found in Smola and Schölkopf [9].

$$\begin{aligned} \text{minimize} \quad & P\left(\mathbf{q}, C, \xi_k^+, \xi_k^-\right) = \frac{1}{2}\mathbf{q}\mathbf{q}^\top + C\sum_{k=0}^{T}\left(\xi_k^+ + \xi_k^-\right) \\ \text{subject to} \quad & \left(z_k - \mathbf{q}^\top\phi(\mathbf{u}_k) - \rho\right) \leq \eta + \xi_k^+ \\ & \left(z_k - \mathbf{q}^\top\phi(\mathbf{u}_k) - \rho\right) \geq \eta + \xi_k^- \\ & \xi_k^+, \xi_k^- \geq 0 \\ & C > 0 \end{aligned} \tag{4}$$

C and η are user specified regularization and precision parameters respectively. They are chosen according to the practical guidelines set forth in [10]. ξ^+, ξ^- are non-zero slack variables, $\mathbf{q}$ is the weight vector normal to the separating hyperplane, ρ is the offset parameter, $\phi(\mathbf{u}_k)$ represents the transformed image of $\mathbf{u}_k \in \mathbb{R}^2$ in the same Euclidean space, and $k \in [0, \dots, P]$. In this paper we have used the RBF (Radial Basis Function) as the mapping function given in Eqn. 5, where σ represents the hyperparameter of the Gaussian function.

$$\langle \mathbf{u}_i, \mathbf{u}_k \rangle = \exp\left(-\frac{1}{2}\frac{\parallel \mathbf{u}_k - \mathbf{u}_i \parallel^2}{\sigma^2}\right) \tag{5}$$

The hyperparameter σ, also known as the "kernel width" parameter, controls the overall scale in horizontal variations. More details on SVR can be found in other work [11].

2.3 k-nearest neighbor regression (k-NN)

k-NN, or k-nearest neighbor regression is most useful for mapping data that is often represented in high-dimensional spaces to lower dimensional manifolds. As such, it is often used for contrast to linear dimensionality reduction techniques such as PCA (Principle Components Analysis). In

fact, this regression technique may be very well suited to the task at hand, as it is well known that fluid flow governed by the nonlinear Navier-Stokes partial differential equations contain infinite degrees of freedom. Techniques for representing such data in lower dimensional nonlinear manifolds have also been studied and well understood for some time now [12] through the Lorenz model. Although the input space spanned by the $(phase, amplitude)$-tuple is limited to two dimensions, it would still explicitly benefit from the implicit dimensionality reduction capability that k-NN regression offers. The $(phase, amplitude)$-tuple can be considered as the latent points defining the low-dimensional representation of the data space (the shear layer thickness), which contains high dimensional patterns. k-NN regression assumes that points in the data space that are located in close proximity to each other have similar output values (*i.e.* the shear layer thickness). As such, for novel $(phase, amplitude)$-tuples presented to the regression algorithm, the output values must be located in close proximity to those k nearest points having similar patterns in higher dimensional space, and so the hyperparameter used for k-nn (nearest neighbor) regression is the number of nearest neighbors. More details on this regression method can be found in [13].

2.4 ℓ_1 regularized linear regression (Lasso)

Another regression method to be investigated is a regularized sparse linear regression method known as LASSO (least absolute shrinkage and selection operator) [14]. Its hyperparameters are already implicitly optimized as part of its algorithm. A valuable benefit which can be derived from using lasso regression is that it can be used to find influential variables due to the nature of its ℓ_1 sparsity "regularization" penalty. Sparseness penalizes the number of non-zero coefficients associated with the linear regression, translating to a "sparse" solution. Due to the nature of the inequality constraint associated with the regularization penalty, the solution is achieved by appealing to the use of cyclical coordinate descent in an iterative fashion, which yields a regularization path of candidate solutions until the algorithm runs to completion. It has been shown [15] that lasso regression is equivalent to a simple linear correlation analysis, which can be used to select the linearly influential variables as well. However, this feature is not very relevant for our case in which there are only two independent variables: magnitude and phase, and thus there is no need for feature selection. However, we use it to contrast among the other regression techniques described in this section.

2.5 Bagged neural nets (BNN)

Artificial neural networks are a well known machine learning technique that was popularized decades ago, and has recently seen a resurgence through a different incarnation which has been called "deep learning." In this study we will only consider traditional neural networks with a single layer perceptron as distinct from "deep learning" techniques which exploit the use of multiple hidden layers. Fundamentally, neural networks offer the ability to capture nonlinearities in data by learning weights associated with nodes in a network that are linearly combined and ultimately transformed through a nonlinear mapping. More details on neural networks and the bagging, or bootstrap aggregating process which aims to aggregate ensembles of neural networks trained on datasets generated from the same source can be found in [16]. Bagging is meant to address deficiencies in stability and accuracy of the neural net performance and also reduces variance to help prevent overfitting. However, due to the lack of complexity in fitting the data for this problem, a single base model was used and no bagging was necessary. The hyperparameter used for bagged neural nets (BNN) regression is the number of hidden units in a single layer perceptron.

3 Discussion and Results

A comprehensive listing of the results for all regression techniques and their respective near global optima are provided in Table 2. Table 1 describes the hyperparameters used for each regression method. Our findings indicate that nonlinear regression methods: support vector regression (SVR), k-NN (nearest neighbor) regression, and neural networks are far superior to the linear regression methods. This suggests inherent nonlinearity in the data, which is also apparent in Figs. 1-3, where the fit of these best three performing nonlinear regression techniques have been illustrated. The top panel of each of the three figures represent the results of hyperparameter optimization. Each panel respectively illustrates the result of a grid search, showing the NMSE as a function of the number of nearest neighbors (kNN regression), the number of hidden units (BNN regression), and the the

kernel width, σ (SVR). Both the optimized function and hyperparameter values are also shown, mirroring the results shown in Table 2.

Table 1: Tunable Regression Hyper-parameters

Regression Method	**Regression Description**	**Hyper-parameter**	**Hyper-parameter description**
SVR	Support Vector Regression	σ	Kernel Width
k-NN	k-NN Regression	k	Number of nearest neighbors
LR1	Ridge (linear) regression using linear regressors	λ	ℓ_2 regularization coefficient
LR2	Ridge (linear) regression using quadratic regressors	λ	ℓ_2 regularization coefficient
LR3	LASSO (linear) regression	λ	ℓ_1 regularization coefficient
BNN	Bagged Neural Networks	n_h	number of hidden units

Table 2: Regression Results

Optimization Results	**SVR**	**k-NN**	**LR1**	**LR2**	**LR3**	**BNN**
Optimized Hyperparameter Value	1.12	2	7.9	376.5	5.74×10^{-5}	4
Optimized Function Value†	0.125	0.14	2.43	3.64	1.054	0.115

† For the objective function shown in Eqn. 1

The middle panels of each of Figs. 1-3 are identical, and illustrate a contour-filled level set view of the actual shear layer thickness as a function of the phase and amplitude. The color bar to the right of the panel indicates the thickness. Finally, in order to gain a qualitiative appreciation for the results quantified in Table 2, the bottom panel of each figure provides a similar contour plot of the surface resulting from application of the respective regression technique. This plot provides an estimate of the shear layer thickness as a function of the perturbation phase and amplitude for each respective nonlinear regression technique, which offers the ability to perform a qualitative comparison to the actual shear layer thickness in the panel directly above it.

Figs. 4(a,b) show how the shear layer thickness has evolved over simulation time of 2.5 corresponding to first pairing of two large structures in the free shear layer, with the phase shift ϕ as a parameter. Fig. 4(a) represents one scenario and Fig. 4(b) represents a second scenario out of many, based on two different forcing levels (amplitudes). The shear layer thickness contour plot shown in Figs. 1, 2 and 3 represents results corresponding to all the forcing levels and the phase shifts of perturbations used in the CFD simulation at a simuation time of 2.5 seconds. Fig. 4(a) shows results coresponding to the forcing level of subharmonic,A_s, being half that of the fundamental, A_f, and Fig. 4(b) shows

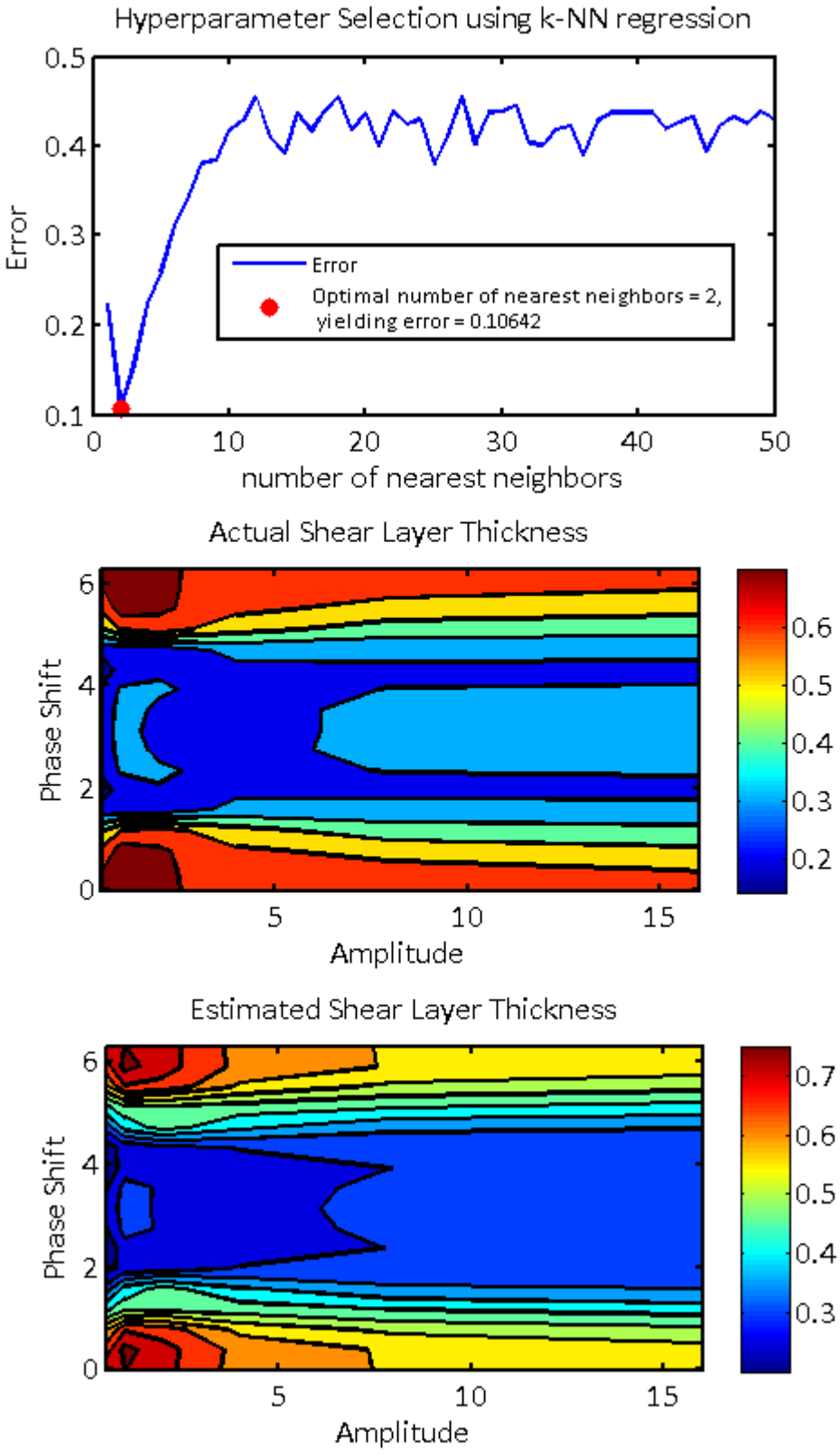


Figure 1: Shear Layer Thickness k-NN Regression Results

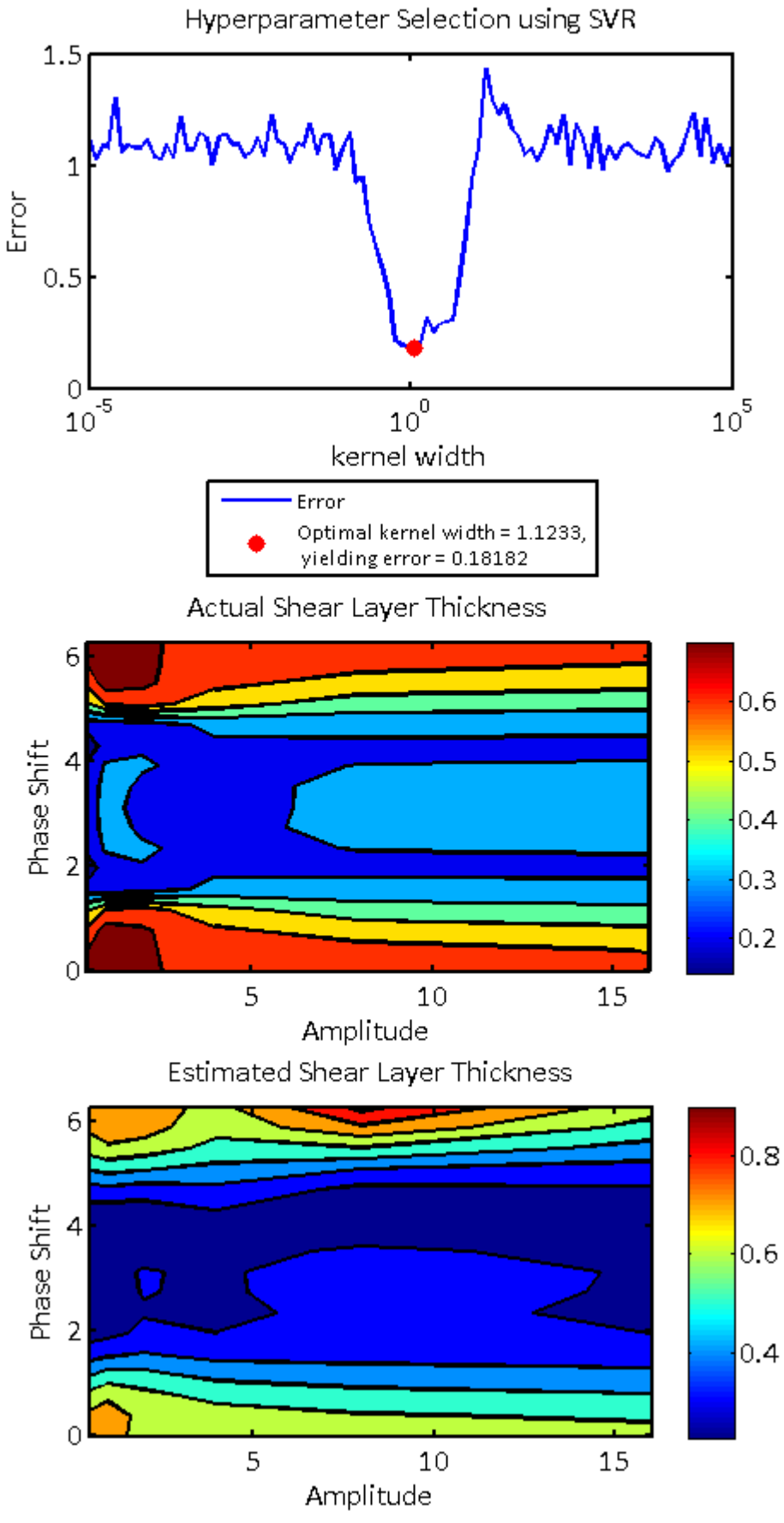


Figure 2: Shear Layer Thickness SVR Regression Results

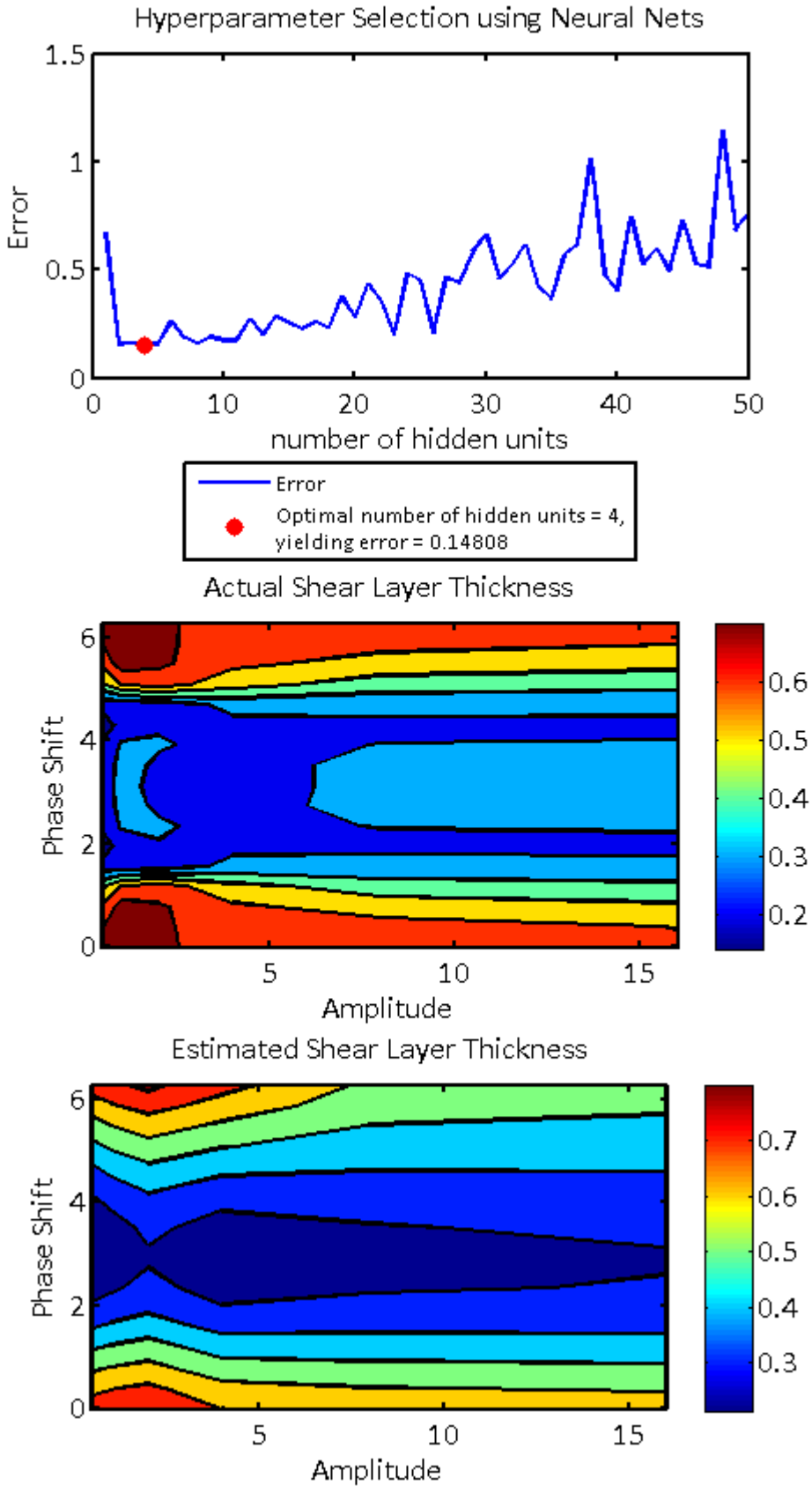
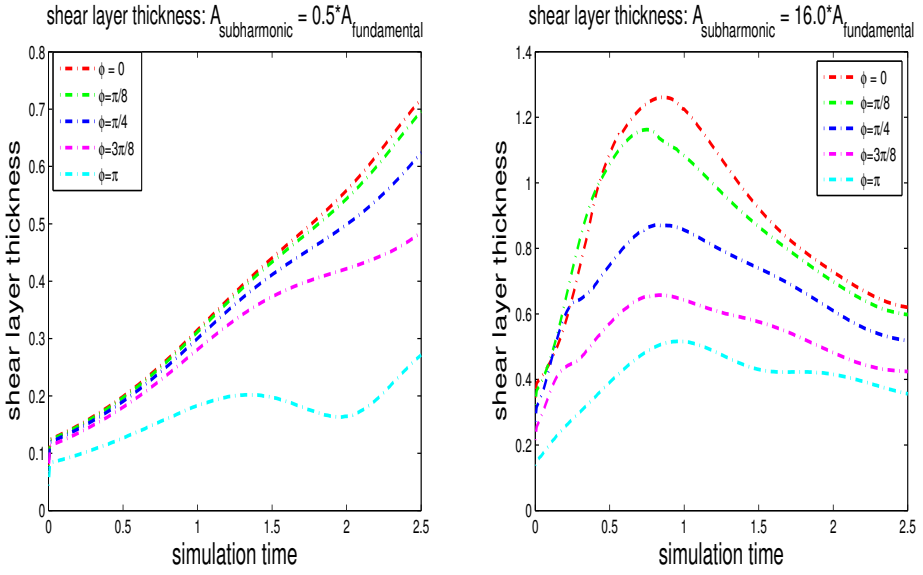


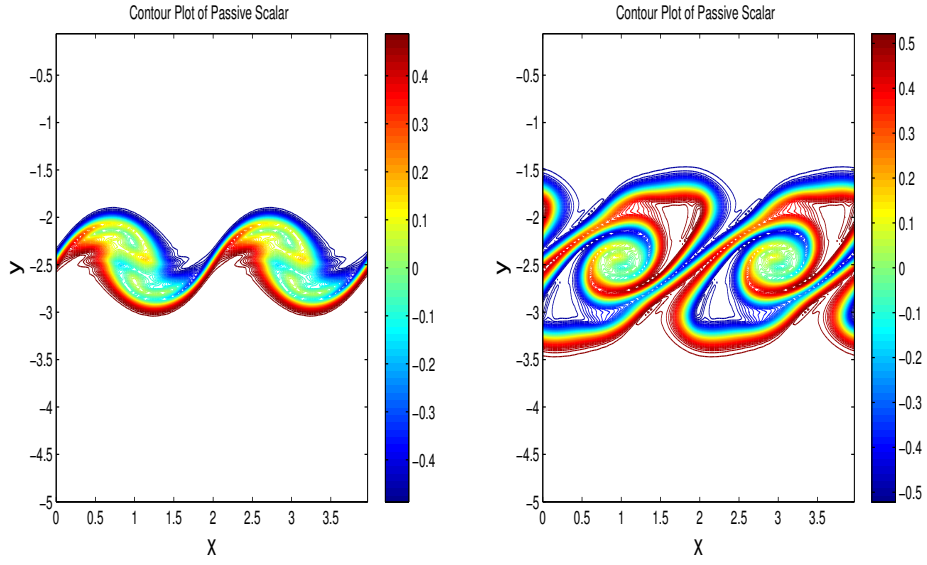
Figure 3: Shear Layer Thickness BNN Regression Results



fourier analu

(a) $A_s = 0.5 * A_f$ (b) $A_s = 16.0 * A_f$

Figure 4: Evolution of Shear Layer Thickness With Simulation Time



(a) $A_s = 0.5 * A_f$ (b) $A_s = 16.0 * A_f$

Figure 5: Instantaneous Passive Scalar Contours

results corresponding to A_s = 16.0*A_f. Fig. 4(a) shows a linear growth upto simulation time of 2.5, while Fig. 4(b) shows that nonlinear growth sets in much earlier.

Figs. 5(a,b) show the passive scalar contour plots corresponding to Figs. 4(a,b), respectively, which demonstrate a marked difference in the extent of mixing between the two forcing levels.

Finally, a comparison of run-time for both training and testing all regression modes is provided in Table 3. It is clear that the testing time for all regression methods for one set of amplitude and phase shift is on the order of milliseconds, as measured on a PC. Due to the fundamental nature of the algorithm, training and testing are executed simultaneously for the k-NN regression method, and as such the testing time is blocked out. It should be noted that it takes one CFD simulation

Table 3: Run-Time Results in Seconds

	SVR	k-NN	LR1	LR2	LR3	BNN
Training Time	0.0034	0.0176	0.0128	0.0238	0.001176	0.388
Testing Time	6.82×10^{-4}		4.69×10^{-4}	0.0108	7.4×10^{-5}	0.0066

corresponding to one set of amplitude and phase shift about 0.3 second on a Mac laptop to generate the training data represented in the midlle panels of Figs. 1, 2 and 3. For any other combination of phase and amplitude not reflected in these figures, it will be faster to use the surrogate models to predict the shear layer thickness, rather than the CFD simulation.

4 Concluding Remarks

The resulting NMSE for LASSO and other linear regression-based techniques were unfortunately not small enough to use as proxy for the physics-based Navier-Stokes equations. As such, we have found that the nonlinear regression methods offer the best approximation to the corresponding CFD simulations, both quantitatively and qualitatively. Furthermore, we have demonstrated that using any of the nonlinear (or even linear) regression methods as a surrogate for the CFD simulation based upon the Navier-Stokes equations reduces the computational burden considerably. The appreciable gap in computation time achieved is thus also well below what is currently known to be required for the development of active flow control methodologies based upon reduced order models of the Navier-Stokes equations.

References

[1] Kaul, U. K., "Efficient CFD-Based PID Control of Free Shear Layer Flow," *43rd AIAA Fluid Dynamics Conference and Exhibit*, San Diego, CA, USA, June 2013.

[2] Kaul, U. K., "Proportional-Integral-Derivative Control of Mixing Layers," *7th AIAA Flow Control Conference*, Atlanta, GA, USA, June 2014.

[3] Cordier, L., Noack, B. R., Tissot, G., Lehnasch, G., Delville), J. D. J., Balajewicz, M., Daviller, G., and Niven, R. K., "Identification strategies for model-based control," *Experiments in Fluids*, Vol. 54, 2013.

[4] Pastoor, M., King, R., Noack, B. R., and Tadmor, G., "Observers and Feedback Control for Shear Layer Vortices," *44th IEEE Conference on Decision and Control and the European Control Conference*, Seville, Spain, December 2005.

[5] Pastoor, M., Henning, L., Noack, B. R., King, R., and Tadmor, G., "Feedback shear layer control for bluff body drag reduction," *Journal of Fluid Mechanics*, Vol. 608, 2008, pp. 161–196.

[6] Pressburger, T., Hoelscher, B., Martin, R. A., and Sricharan, K., "Simple Sensitivity Analysis for Orion GNC," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Boston, MA, USA, August 2013.

[7] Bock, J. and Gough, D., "A New Method to Estimate Ligand-Receptor Energetics," *Molecular and Cellular Proteomics*, Vol. 1, No. 11, November 2002, pp. 906.

[8] Chu, E., Gorinevsky, D., and Boyd, S. P., "Detecting Aircraft Performance Anomalies from Cruise Flight Data," *Proceedings of the AIAA Infotech@Aerospace Conference*, Atlanta, Georgia, April 2010.

[9] Smola, A. J. and Schölkopf, B., "A Tutorial on Support Vector Regression," Tech. rep., Statistics and Computing, 2003.

[10] Cherkassky, V. and Ma, Y., "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Networks*, Vol. 17, January 2004, pp. 113–126.

[11] Martin, R. and Das, S., "Near Real-Time Optimal Prediction of Adverse Events in Aviation Data," *Proceedings of the AIAA Infotech@Aerospace Conference*, Atlanta, Georgia, April 2010.

[12] Weigend, A. and Gershenfeld, N., editors, *Time series prediction: forecasting the future and understanding the past*. SFI studies in the sciences of complexity, Addison-Wesley, 1993.

[13] Krämer, O., "Dimensionality Reduction by Unsupervised K-Nearest Neighbor Regression," *IEEE 10th International Conference on Machine Learning and Applications*, Honolulu, HI, USA, December 2011.

[14] Hastie, T., Tibshirani, R., and Friedman, J., "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, Vol. 33, No. 1, January 2010.

[15] El Ghaoui, L., Viallon, V., and Rabbani, T., "Safe Feature Elimination in Sparse Supervised Learning," Tech report ucb/eecs-2010-126, EECS Department, University of California, Berkeley, September 2010.

[16] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, 2009.